



Indo Além do Básico com JavaScript



lá, tudo bem?

Nesta unidade vamos nos aprofundar na linguagem JavaScript, estudando como aplicar esta linguagem com o paradigma de programação orientado a objetos. Com esta técnica de programação, o projeto de software deve se basear no conceito de objetos para modelar tipos de dados abstratos e que serão representados dentro de nossos programas.

Como vimos na unidade anterior, quando utilizamos em nosso código o método document.getElementById() para recuperar elementos HTML do DOM, estamos retornando um objeto do tipo HTML. Isso nos permite acessar e modificar as várias propriedades desse objeto. Ou seja, já temos à nossa disposição vários objetos pré-definidos em JavaScript: aqueles disponíveis por intermédio do DOM e ainda outros que veremos nas próximas unidades, que estão disponíveis via APIs dos navegadores.

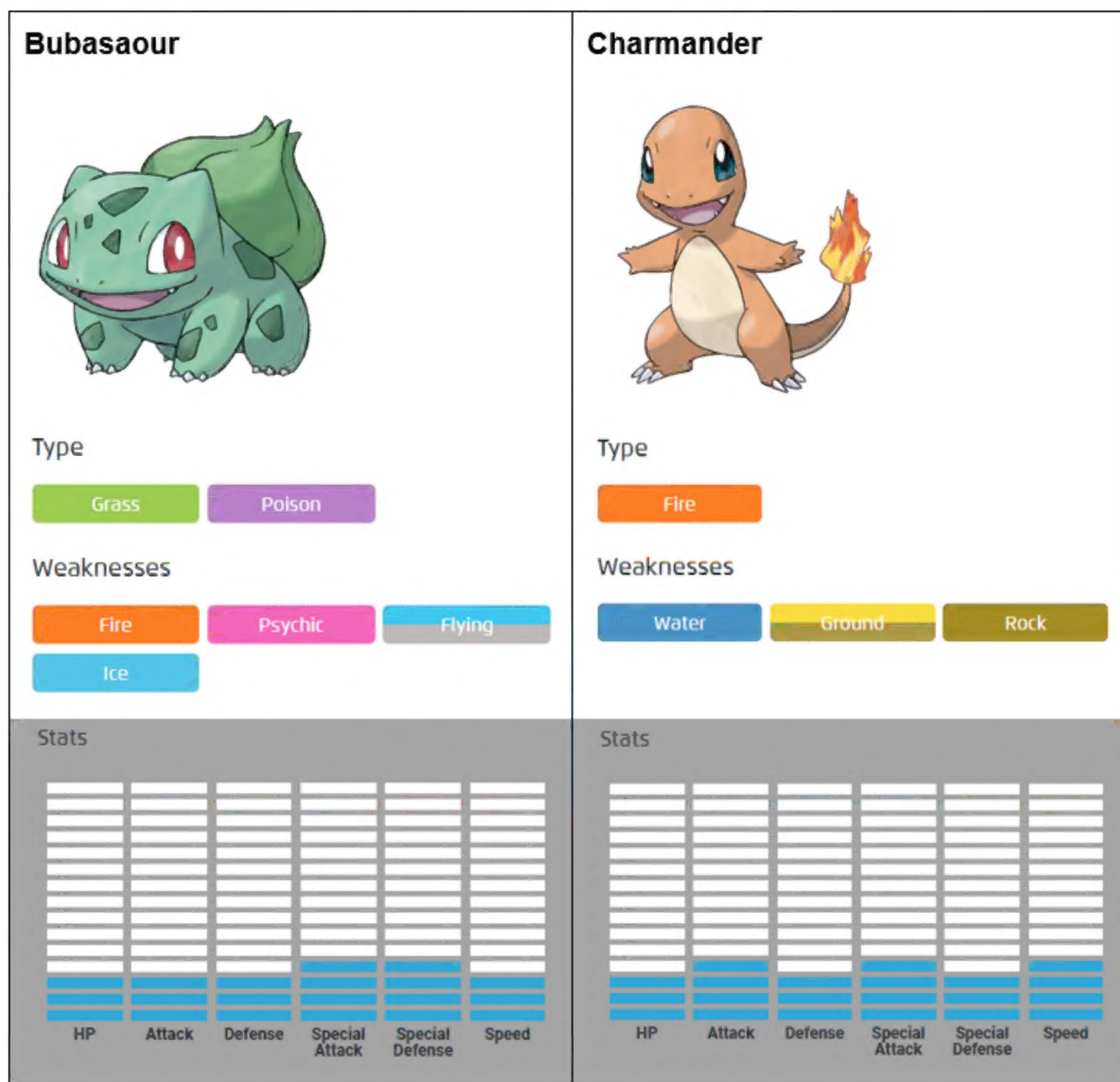
Na verdade, quase tudo em JavaScript pode ser tratado como um objeto, pois possuem todas as características de objetos. Incluindo os tipos primitivos de dados, excetuando *null* e *undefined*.

Breve Visão Sobre Orientação a Objetos

Quando falamos em orientação a objeto, estamos nos referindo ao paradigma orientado a objeto. Ou seja, que nosso pensamento e forma de programar

serão com base nesse paradigma. Iremos nos guiar então nos conceitos de **objetos**, **classes** e **herança**.

Para compreendermos os conceitos básicos do paradigma orientado a objeto, vamos utilizar um exemplo de jogo clássico. Observe a figura a seguir, onde temos dois personagens bem populares.



Podemos verificar que são dois exemplos de Pokémon: o bubasaour e o charmander. Ou seja, com base no paradigma orientado a objetos, podemos dizer que eles são definidos como **objetos** da mesma **classe Pokémon**. Ou seja, temos os objetos **charmander** e **bubasaour**.

Assim, podemos observar que um Pokémon possui várias características que o definem como pertencente à classe Pokémon. Esse conj de características é associado a cada objeto Pokémon e o que difere cada exemplo dessa classe são os seus valores, que são diferentes.

Essas características são denominadas **atributos**. Note que o **objeto charmander** recebe os mesmos atributos e operações da **classe Pokémon** à qual pertence, mas possui diferentes características do **objeto bubasaour** (que também pertence à **classe Pokémon**).

Ou seja, uma classe possui atributos que a identificam; logo, identificam também os objetos dessa classe. Como podemos ver na imagem anterior, um Pokémon possui um **tipo** (*type*), **fraquezas** (*weaknesses*) e outras estatísticas (*stats*) como **pontos de vida** (HP), **pontos de ataque** e **defesa**, **velocidade**, etc.

Então, aqui temos mais um conceito: **herança**. Em orientação a objeto, herança é quando um objeto carrega consigo os atributos e métodos da classe à qual pertence. Nesse exemplo, todo objeto da classe Pokémon pode ser manipulado de várias formas em nosso programa. Um objeto dessa classe pode ser capturado, usar uma habilidade especial ou fazer um ataque simples, por exemplo. Cada uma dessas possibilidades é compreendida como uma **operação**. Cada operação será implementada como um **método da classe.**

Um objeto encapsula dados, métodos, outros objetos, constantes e outras informações relacionadas, o que chamamos de **encapsulamento**. Em outras palavras, toda essa informação é **empacotada** sob um nome e pode ser reutilizada como uma especificação do projeto de software em análise (PRESSMAN, 1995, p. 320).

Resumindo, o objeto são os dados, seus atributos e operações que serão manipulados em nosso programa. Uma classe é a categoria na qual os

objetos são organizados ou classificados, que também possuem atributos e operações. Já a herança são as características que um objeto herda da classe mãe.

JavaScript Orientado a Objetos

Em JavaScript, pode-se dizer que um objeto é um tipo especial de variável que contém atributos e métodos. Usar essa abordagem na linguagem JavaScript evoluiu ao longo do tempo e podemos utilizar esse paradigma de programação de várias formas.

Sendo assim, pode-se lançar mão de uma maneira mais simplificada ou buscar usar todos os conceitos do paradigma de orientação a objetos, num nível mais avançado, trabalhando com classes, heranças e polimorfismo. Essa escolha dependerá do projeto do seu jogo web e do nível de complexidade que ele terá.

Objetos na Notação Literal

Podemos criar objetos utilizando a notação literal, por meio de um inicializador de objetos. A sintaxe de criação objetos dessa forma usa uma notação de chave: valor dentro de { }. Assim para criar o objeto charmander seria o seguinte:



```
// Notação literal para declarar um Objeto
const charmander = {
  // Atributos do Objeto
    hp: 3,
    attack: 4,
    defense: 3,
    speed: 4,
    type: "Fire",
```

```
    weaknesses: ["Water", "Ground", "Rock"]
  // Métodos do Objeto
  toStrike: function(target) {
      // Bloco de código para executar o ataque
    // target seria o alvo do ataque
    },
    toGuard: function() {
      // Bloco de código para ação defensiva
    }
};
```

Como é possível observar os atributos do objeto recebem valores ou objetos. Já os métodos são declarados como a palavra-chave function.

Então, para que o objeto charmander execute a operação de ficar na defensiva, escreveríamos a seguinte linha de código:

```
charmander.toGuard();
```

Esse formato literal de declaração é útil para quando tivermos somente um objeto de uma determinada classe.

Assim, para declarar os objetos charmander e bubasaour como pertencentes à classe Pokémon, devemos usar a abordagem de função construtora.

Classes com a Função Construtora

Essa abordagem é uma alternativa para criarmos classes de objetos em JavaScript. Para tal, criamos uma função para a classe de objeto que especifique seu nome e dentro dela inserimos suas propriedades e métodos.

```
// Função Construtora para declarar uma classe
function Pokemon(_hp, _attack, _defense, _speed, _type) {
    // Atributos da classe
    this.hp = _hp;
    this.attack = _attack;
```

```
    this.defense = _defense;
    this.speed = _speed;
    this.type = _type;
    // Métodos do Objetos
    this.strike = function (target) {
      // Bloco de código para executar o ataque
    // target seria o alvo do ataque
    }
    this.guard = function () {
      // Bloco de código para ação defensiva
    }
}
```

Note que devemos utilizar a palavra-chave this para atribuir valores aos atributos da classe com base nos valores passados como argumento da

função. Dessa forma, estamos declarando as variáveis dos atributos da classe.

Sendo assim, para criar instâncias da classe utilizamos a palavra-chave new. Por exemplo, para criarmos os objetos charmander e bubasaour como instâncias da classe Pokémon, faremos:

```
// Criando o objeto charmander, instanciando a classe pokemon
let charmander = new Pokemon(3, 4, 3, 4, "Fire");
// Criando o objeto bubasaour, instanciando a classe pokemon
let bubasaour = new Pokemon(3, 3, 3, 3, "Grass");
```

Então, para que o objeto charmander execute um ataque ao bubasaour, escreveríamos a seguinte linha de código:

```
charmander.strike(bubasaour);
```

Pronto! Agora já podemos criar jogos de uma forma muito mais prática e inteligente. Os conceitos trabalhados nessa unidade e a sua aplicação em programação exige prática. Por isso, não deixe de realizar as atividades práticas e confira o que deixei para você em atividade extra!

Bons estudos e fortes abraços!

Atividade Extra

O site da MDN Web Docs possui um artigo excelente (e todo em português) sobre orientação a objetos com JavaScript. Leia este material para completar e fixar os seus conhecimentos dessa unidade. Este artigo apresenta uma visão básica da teoria de programação orientada a objeto e explora como o JavaScript emula as classes de objetos através de funções de construtor e como criar instâncias de objeto.

- **JavaScript orientado a objetos para iniciantes**, disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/Object-oriented_JS>.

Acesse o site de exercícios da W3Schools e realize os 3 exercícios sobre objetos em JavaScript.

- **Exercícios da w3schools sobre objetos em JavaScript**, disponível em: <https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_objects1>.

Referência Bibliográfica

COAD, P.; YOURDON, E. **Object-Oriented Analysis.** São Paulo: Prentice-Hall, 1990.

PRESSMAN, R. S. **Engenharia de software**. 7 ed. São Paulo: Makron Books, 1995. Disponível em: <https://pt.scribd.com/doc/257550621/Roger-S-Pressman-Engenharia-de-Software-7-Edicao-Uma-Abordagem-Profissional>.

Ir para exercício